REMARKS

Applicant thanks the examiner for his time and comments during the telephone interviews held on October 2, 2007, and October 29, 2007, and attended by Examiner Dare and applicant's undersigned attorney, Ido Rabinovitch. During the Examiner's Interviews, the claims and the cited art were discussed. No agreement regarding the claims was immediately reached. The examiner did, however, indicate that amendment of claim 1 to include execution of the transformed vector memory access instructions might overcome the current rejections and put the application in condition for allowance.

The examiner rejected claims 1, 4, 6 and 14 under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 6,292,938 to Sarkar et al..

The examiner also rejected claims 2-3, 13, 15-18, 25-26 and 28-30 under 35 U.S.C. §103(a) as being unpatentable over Sarkar in view of U.S. Patent No. 6,571,319 to Tremblay et al.

The examiner further rejected claims 7 and 9-12 under 35 U.S.C. 103(a) as being unpatentable over Sarkar in view of the Microsoft Press Computer Dictionary, and rejected claim 19-24 under 35 U.S.C. 103(a) as being unpatentable over Sarkar and Tremblay, in view of the Microsoft Press Computer Dictionary.

With respect to applicant's claim 1, the examiner stated:

> 3. With respect to claim 1, Sarkar teaches a method comprising:
> converting memory access instructions into intermediary standard formatted memory access instructions, in col. 4, lines 54-60.
> generating a plurality of memory access partitions containing corresponding subsets of the intermediary standard formatted memory access instructions, with the plurality of memory access partitions directed to specific memory banks, in col. 2, lines 56-60.
> identifying matching instructions based on comparisons of pre-defined instruction patterns to the intermediary standard formatted memory access instruction in the plurality of memory access partitions, in col. 2, line 64 through col. 3, line 14; and
> transforming the matches to vector memory access instructions, in col. 3, lines 15-29. (Office Action, page 2)

Applicant amended independent claim 1 to clarify that the transformed vector memory access instructions, when executed, cause a corresponding memory access operation to be

performed for a group of memory locations. Support for this clarification is provided, for example, at page 5, line 25 to page 6, line 5, of the originally filed application.

Sarkar describes optimizing compilers for computer programs that perform retargeting optimized code by matching tree patterns in directed acyclic graphs (col. 1, lines 7-10). Sarkar explains:

> **The solution provided by the present invention is based on a two-level partitioning. An optimized basic block DAG is first partitioned into trees that contain only true (data) dependence edges and in which each node has at most one true dependence out-edge. Such a tree of true dependences is called a fan-in tree. Each fan-in tree is then supplied as input to tree pattern matching. The output of tree pattern matching is a decomposition of the fan-in tree into patterns, which is represented as a second-level partition of each fan-in tree into subtrees. The present invention's goal for DAG partitioning is to find a minimum cost legal two-level partition of the entire DAG into subtrees. The present invention gives precise conditions based on data dependence theory [25] for identifying legal two-level partitions and provides an efficient greedy method as a heuristic solution. The present invention also extends this efficient greedy method to selectively perform duplication of IL instructions to further improve the quality of the target code.**
>
> **In addition to retargeting optimized code in static compilers, it is anticipated that an important future application of the present invention's approach will be in retargetable code generation for mobile code systems, such as Java. A key requirement for mobile code is that it have a machine-independent intermediate form that can be conveniently translated/executed on different target processors. Retargeting the intermediate form becomes a greater challenge for larger sets of target processors, especially in embedded systems where there is a great proliferation of target processor instruction sets. However, the approach described in this application for retargeting optimized code by matching tree patterns in DAGs can be used to quickly build translators from virtual machines to several different target processors. (Col. 2, line 54, to col. 3, line 29)**

Sarkar also explains:

> **FIG. 2 illustrates the structure of an exemplary source translator 116 (e.g., a compiler) that could be used to implement the preferred embodiment of the present invention.**
>
> **Source code 122, such as C or C++, is accepted by a compiler front end 200 and then translated to an intermediate language (IL) by a optimizing back-end 202. C and C++ were chosen as the programming languages in this example, although the structure shown in FIG. 2 can also be used to build source translators for other languages, such as Fortran, Java, PL/1, Cobol, etc.**

> **A machine grammar 204 is fed into a modified, bottom-up, rewrite generator (MBURG) 206 to obtain a set of pattern matching tables for a specific target processor. In addition, machine-specific parameters (e.g., description of register sets for the target processor) are fed into both the optimizing back-end 202 and a retargetable code generator 208.**
>
> **The retargetable code generator 208 implements the partitioning and duplication methods outlined in Sections 4 and 5 below. Its input is the IL that is generated towards the end of the optimization steps performed by the optimizing back-end 202. In the preferred embodiment, this point is just after global register allocation. This means that, even though the input IL to the retargetable code generator 208 is architecture-neutral (i.e., it does not reflect a specific target instruction set), a target-processor-specific global register allocation has already been encoded in the structure of the IL. If the retargetable code generator 208 is used to translate mobile code, then the IL should be sent to the retargetable code generator 208 at a point that precedes global register allocation.**
>
> **The output of the retargetable code generator 208 is object code 124 for the target processor that is generated by performing tree pattern matching (Step 314). Those skilled in the art will recognize that the object code 124 may comprise either assembly code or binary code, as desired. (Col. 4, line 50, to col. 5, line 19)**

However, at no point does Sarkar describe transforming any type of instruction or code (source or intermediary) into instructions, such as vector memory access instructions, that, when each such instruction is executed, causes a group of memory locations to be accessed. Accordingly, Sarkar fails to disclose or suggest at least the features of "transforming the identified matching instructions to vector memory access instructions, the transformed vector memory access instructions, when executed, causing a corresponding memory access operation to be performed for a group of memory locations," as required by applicant's independent claim 1. Independent claim 1 and the claims depending from it are therefore patentable over the cited art.

As noted, the examiner rejected independent claim 15 as being obvious over Sarkar in view of Tremblay. Independent claim 15 recites includes the feature that the transformed vector memory access instructions, when executed, causing a corresponding memory access operation to be performed for a group of memory locations."

The examiner admitted that "Tremblay teaches the above limitations but does not teach the last two limitations of claim 15" (Office Action, page 5). It follows, therefore, that Tremblay fails to teach "transforming the intermediary memory access instructions in the subsets

corresponding to the plurality of memory access partitions that match pre-defined instruction patterns to vector memory access instructions, with the transformed vector memory access instructions, when executed, causing a corresponding memory access operation to be performed for a group of memory locations."

The examiner, however, relied on Sarkar as allegedly teaching the features pertaining to the last two recited features of applicant's claim 15, including the feature "transforming the intermediary memory access instructions in the subsets corresponding to the plurality of memory access partitions that match pre-defined instruction patterns to vector memory access instructions, with the transformed vector memory access instructions, when executed, causing a corresponding memory access operation to be performed for a group of memory locations". For reasons similar to those provided with respect to independent claim 1, applicant contends that Sarkar does not disclose or suggest at least the features of "transforming the intermediary memory access instructions in the subsets corresponding to the plurality of memory access partitions that match pre-defined instruction patterns to vector memory access instructions, with the transformed vector memory access instructions, when executed, causing a corresponding memory access operation to be performed for a group of memory locations," as required by applicant's independent claim 1.

Because neither Sarkar, nor Tremblay, discloses or suggests, alone or in combination, at least the above features applicant's independent claim 15 and the claims depending from it are therefore patentable over the cited art.

Independent claim 27 recites "transform the identified matching instructions to vector memory access instructions, with the transformed vector memory access instructions, when executed, causing a corresponding memory access operation to be performed for a group of memory locations". The examiner did not indicate on what ground, if at all, independent claim 27 was rejected. However, the examiner rejected claim 28, which depends from claim 27, as being obvious over Sarkar in view of Tremblay. Even assuming, therefore, that independent claim 27 may have been rejected as being obvious over Sarkar in view of Tremblay, applicant contends that for reasons similar to those provided with respect to independent claim 15,

applicant's independent claim 27 and the claims depending from it are patentable over the cited art.

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

In view of the foregoing, applicant respectfully submits that the application is in condition for allowance and such action is respectfully requested at the examiner's earliest convenience.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

Please apply any required fees to deposit account 06-1050, referencing the attorney docket number shown above.

Respectfully submitted,

Date: Nov. 2, 2007

Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

PTO Customer No. 20985
Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906